Grif: An Interactive System for Structured Document Manipulation

Vincent Quint*, Irène Vatton**
**INRIA*
***CNRS*

ABSTRACT
Grif is an interactive system for editing and formatting complex documents. It manipulates structured documents containing objects of various types: tables, mathematical formulæ, programs, pictures, graphics, etc... It is a structure directed editor which guides the user in accordance with the structure of the document and of the objects being edited; the image displayed on the screen also being constructed from that structure. Flexibility is one of the most interesting characteristics of Grif. The user can define new document structures and new types of objects, as well as to specify the way in which the system displays these documents and objects.

1. Presentation

Existing document manipulation systems may be classified into various categories. There are batch formatters [Furuta] and interactive systems [Meyrowitz]. Some formatters such as Scribe [Reid] or Mint [Hibbard] consider the logical structure of the documents they manipulate. Some others, like TEX [Knuth] or Troff [Kernighan], are more concerned with layout, even if macros allow some structure to be introduced in the document.

Formatters have also evolved towards more friendly tools, that allow the user to see quickly on the screen the result of his work: TEX, for example, has several 'preview' systems. Janus [Chamberlin] is an original system that has been developed with the same approach. Although they allow the user to see the final form of the document on the screen, these systems cannot be considered as really interactive, as they do not allow the user to interact directly on the final form of the document. Other extensions to formatters have been proposed, by adding a truly interactive editor [André].

Interactive systems may also be classified as layout oriented or structure oriented, but structure oriented interactive systems are mainly used for a particular class of documents: programs. Mentor [Donzeau], the Cornell

Program Synthesizer [Teitelbaum], Gandalf [Medina] or Adèle [Estublier] have been primarily developed for programming environments.

Interactive or not, all systems accept text. Some of them accept other forms of information, like Troff with its specialised preprocessors for mathematics, tables, and graphics. TEX, Scribe and Mint accept mathematics and tables. Some interactive systems also manipulate mathematics [Quint], or graphics [Joloboff83], or both [Joloboff84] in a structured way.

Finally there are interactive systems designed with an agreeable user interface which allows text, tables and graphics to be manipulated. The first commercially available system of this kind was the Star [Harslem]. The same principles have been used in the Macintosh or in Interleaf WPS, but the structure taken into account in these systems is limited.

An interesting synthesis of all these systems has been realised with Andra [Gutknecht]. We propose to go further in that direction with Grif, by trying to put together the most interesting features of these various kinds of system. Grif uses the structure of the documents it manipulates; it accepts text as well as other forms of information; it is really interactive and flexible, allowing the user to define his own structures and layouts and it presents a modern user interface with mouse, windows, menus, variable size characters, etc...

2. Document Structure

2.1. The Logical Structure

Documents, and specially scientific and technical documents, frequently include objects of various types: mathematical formulæ, tables, photographs, programs, etc... Most of these objects are logically structured. As an example, a mathematical formula may be considered as a sequence of expressions such as fractions, integrals, or summations. A fraction contains a numerator and a denominator which are themselves expressions. An integral contains an integrand and two limits, which are also expressions. A summation or any other mathematical expression may be described in the same way.

We call that the *generic structure* of mathematical formulæ; any formula can be built following that structure which defines the class 'formula'. A *class* is a set of objects built with the same generic structure. Each object of a class has a *specific structure* which conforms to the generic structure of the class and decribes the logical organisation of that particular object.

As well as formulæ, tables may be considered from a logical point of view. Some tables are constituted by a heading and a sequence of lines; each line containing a title and a sequence of cells. This simple model defines a class 'table1'. Other tables may be considered as a sequence of columns, each one having a title and the same number of cells. This generic structure defines another class, 'table2'. It is also possible to define a more general generic

structure which encompasses these two kinds of organisation and defines a unique class 'table'. In Grif, we allow the user to choose the level of generality for his generic structures.

Programs also may be structured in the same way, the generic structure being then directly derived from the grammar of the programming language. Pictures and photographs, as opposed to formulæ, tables and programs, are often considered with a very poor structure: an array of pixels.

In addition to these structures, a general structure organises the document for the objects included therein. For example, a report comprises a title, the author's name, an abstract, a sequence of chapters and a bibliography. Each chapter contains a title and a sequence of sections, and so on. This may be the generic structure for the class 'report'. Other classes of documents, like letters or contracts, have different generic structures.

While creating or editing a document, the author has this kind of structure in mind. If he uses a system that knows about that structure, he can express himself in his own terms and the system can help him in building a correct document, i.e. a document which conforms to the generic structure of its class.

The logical structure of the document helps the user, but also the system. Most functions of the system are based on the structure. Numbering is one of these functions. In a complex document, chapters, sections, footnotes, figures and bibliographic quotations may be automatically numbered by the system, just by using the structure. Moving across the document is another function based on the structure. The system may provide the user with commands, as for example "move to the next chapter", or "to the previous table".

As several generic structures are simultaneously needed for editing a complex document, the higher level structure is called the *document structure*, the others being called the *objects structure*. Grif does not make any difference between a class of documents and a class of objects. The same generic structure may be used for editing an isolated table (for example when preparing a slide) or a table within a document. The class 'paper' may be considered as a class of documents by the author of a paper submitted to a conference, or as a class of objects by the proceedings editor.

Structures manipulated by Grif are trees. The main structure is a tree representing the whole document, but most objects included in the document are sub-trees of that tree. Each node of the tree has a type (title, chapter, numerator...) and may have *attributes* which add semantics to the element. Examples of attributes are the language in which a part of the document is written, or the importance of a fragment of text (keyword or index). These attributes are useful for several different applications working on the document. An information retrieval system uses keywords, a formatter hyphenates words according to the language.

In addition to the main tree structure, there are *references* which represent non-hierarchical relationships between elements; this allows an element to be referenced from another independently of their relative positions in the tree. It is thus possible from any paragraph to refer to a chapter ("As seen in chapter X..."), a figure or a bibliographic quotation. The system may then substitute the number of the referenced element or its title or any other of its sub-elements for the reference.

2.2. Presentation Based on Structure

Using this kind of structure, the system can display or print the document with very little additional information from the author. The presentation of a document or object may be automatically generated from the structure. For example, a fraction is displayed with the numerator centered over the denominator, with a horizontal bar between the expressions; the title of a report is printed with large letters and centered in the page.

If the system knows about the structure of the objects it manipulates and if it has rules for presenting these structures, the user does not have to worry about presentation: when the title of a report is entered, the system can automatically change the character size and center the text.

Another advantage of this automatic presentation is that it leads to a homogeneous presentation for all documents belonging to the same class. All reports will have their title centered and written with the same size, if the system uses the same presentation rules for the class 'report'. The presentation is not intermixed with the structure or the content of the document. It is described outside the document, within the set of presentation rules associated with the class. This set of rules is called a *presentation schema.*

Several presentation schemas can be defined for the same class of documents, allowing the user to display the document in different ways according to the task to be performed. It is thus possible to use a simple presentation schema for editing and a more sophisticated one for printing, without modifying the document itself, just by changing the presentation schema used by the system.

The presentation schema defines how each type of element defined in the structure schema is to be displayed. There are presentation rules for elements of the structure and for attributes.

In a presentation schema, several views may be defined. As in PECAN [Reiss], a *view* is a partial representation of a document. So, a table of contents is a view which collects together all section titles. The presentation schema allows any number of views to be defined, specifying, for each type of element, in which views the element is visible and how it is to be presented in those views. While editing, the user can create the views he requires, being able to destroy any view at any time.

As the layout of the document can be built from its structure and from a presentation schema, a document is stored with only its specific structure and its content. This representation is called the *external representation.*

2.3. A Flexible Structure

Structure oriented systems are often blamed for their rigidity, especially in the field of textual document manipulation. They are well accepted for editing programs, because, for a given language, programs have a fixed and well known generic structure; but the structure of documents is not so stable and it is impossible to describe all possible documents with a unique generic structure.

Grif solves this problem by accepting different generic structures for different classes of documents. In addition, these classes are not fixed; the user may specify new generic structures by writing a variety of programs, called *structure schemas.* A language has been developed for this purpose which resembles somewhat the type declaration part of Pascal.

Flexibility is also enhanced by the possibility of using different classes of objects simultaneously, for the same document. The generic structure specified in the structure schema may impose the type of certain parts of the document, but in most cases, as in a paragraph for example, the user is free to choose the type of the element: pure text, a mixture of text and formulæ, a table, or any other content.

3. Architecture

Another originality of Grif comes from its architecture. Grif's end user interface functions are separated from the processing functions. The former is a specialized software called the Mediator while the latter belongs to the Editor. The Editor checks the specific structure of documents; it elaborates them according to their generic structure and prepares their display. The Mediator takes full responsibility for the physical support; it produces pictures on the screen and manages all physical interactions.

The main reasons for separating Grif's two components are :

Device independence: Grif must be able to interact with its user by means of a standard terminal as well as on a highly sophisticated workstation. Using up-to-date graphic technology and interactive input devices it can provide a versatile user interface, but document manipulation features do not have to change with the terminal. Only the user interface is device dependent.

Customisation: Designing the dialogue for an interactive application is a complex task although designers can now be helped by tools developed for design of dialogue scenario [Wong]. By offering a general tool for physical interaction management, the Mediator allows the Editor to focus on its dialogue. Concurrently, the Mediator realizes the customisation of the user

interface. It can generate a more or less verbose information exchange with the user for a single Editor command.

Man-machine interface prototyping: The Mediator is an embryonic man-machine interface package. It does not yet present all the generality to be expected for such a product but offers some interesting characteristics. Most of the existing user interface managers [Kasik, Harslem, Gosling] are very close to device functionalities, describing terminal behaviour rather than user behaviour. With these interface managers, applications usually precisely control the positions of the drawing on the screen and sometimes use virtual function keys as if they were using a virtual terminal. In Grif, exchanges are at a more logical level. The Mediator is seen as just one of the Editor's potential users; another user could be an application able to execute automatic updates in documents using the same exchange level with the Editor. Exchanges between the Mediator and an application program are based on only two notions : abstract pictures and commands.

3.1. Operation

The Editor uses document and objects structure schemas for generating the abstract representation of a document. The *abstract representation* is a tree which represents the whole document. Using presentation schemas in accordance with structure schema requirements, the Editor produces an *abstract picture* of the document. An abstract picture gives the logical layout of what the Mediator has to show. It is in fact a tree describing the arrangement of several types of units: texts, pictures, graphics and symbols. These units are the leaves of the tree. A node of an abstract picture corresponds usually to a node of an abstract representation, but occasionally several nodes are added within abstract pictures. These additional nodes correspond to presentation elements. The Mediator analyses each abstract picture and, by taking into account the physical characteristics of the device (character size, window size ...) it constructs a *set of boxes* making up the real picture, which is finally displayed.

Concurrently, the Editor defines the set of *commands* it will be able to interpret. The Mediator builds forms and menus to present these commands to the end user and returns the user's answers to the Editor.

3.2. Picture Display

A node of an abstract picture gathers together information and *presentation constraints* which define how the associated information is to be presented. The Mediator builds a *box* for each node of an abstract picture; the notion of box being the same as that used in TEX. Presentation constraints permit the calculation of the width, height and positioning of the boxes, the bodysize and style of the characters and the construction of lines.

In all cases, presentation constraints are completely device independent. So the choice of font style and size is performed by the Mediator by deduction from two logical constraints: the highlight level and the relative pointsize. In the same way, the horizontal and vertical units used are converted into physical units according to the associated relative pointsize. The construction of lines is performed by the Mediator but constraints act on their characteristics: adjustment, centering, indentation, length, vertical spacing...

Finally, the set of produced boxes is organized into a connected graph which represents all relationships between boxes (relative positions or dimensions). The connected graph allows the Mediator to update the real picture as soon as the document content is modified.

3.3. Command Display

Due to the design, the Mediator does not know about the semantics of commands. Actions as well as action parameters are commands for it, but classes of commands are defined so as to allow it to do syntactic verification. Knowing the expected answer layout, it can check the user input, react rapidly to errors and choose the right tool for data acquisition. The different classes available are menu, text, numerical value and designation of the whole or a part of an abstract picture. The Mediator also permits these different classes of commands to be intermingled, in which case it controls the sequence of data acquisition and returns command anwers when the user input is completed.

Concurrently, the Mediator itself performs text editing when it acquires command answers as well as when it updates text into documents. To do so, it predefines four commands: Copy, Delete, Insert, Undo (see figure 1) which operate on strings.

4. Interactivity

Until now, interactive structure-oriented systems have mainly been used for manipulating programs. They have been designed to be used by computer-oriented people. The structure of the program being edited is represented within the system by a tree and the user is supposed to have a good knowledge of this structure as commands are often expressed by reference to that tree. This approach cannot be used in a document manipulation system where the user is unfamiliar with such concepts. Although a rich structure is necessary for powerful processing, it is not desirable to use it systematically in the dialogue between the user and the system.

Grif is an interactive system: every user action immediately reacts on the document content and document change is directly visible to the user. Of course, the Editor takes advantage of its knowledge of the documents being handled. It knows their structure and so can conduct the user in the elaboration

and modification of the document. So it offers, like syntactic editors used in programming environments, powerful commands based on structure, especially for movement, selection and creation.

If on one hand structural organisation of documents permits some manipulation, it complicates normal text manipulation. In Grif, structure manipulation does not inhibit text manipulation. The user can select a part of the document by searching for character strings, extend this selection by pointing to a character as well as by invoking a structure selection command. When the user deletes, copies, inserts or undoes within a part of the displayed picture of the document, Grif manages the right choice between manipulations done by structure and those done by text. The Mediator manipulates the text part of the document and calls the Editor for updating the structure part.

5. An Example

In this section we present an example of a document processed by Grif. For the sake of brevity, this example is a simplification of a real document. Figure 1 is a "hard-copy" of the screen that shows what the user actually sees while editing the document **Example.**

5.1. The Generic Structure

This document belongs to the class **Report,** whose generic structure is defined by the following structure schema.

```
Report = BEGIN
         Title = Text;
         Authors = LIST OF (Author = Text);
         Abstract = LIST OF (Paragraph);
         Body = LIST OF (Section =
                              BEGIN
                              Section-Title = Text;
                              Section-Body =
                                        BEGIN
                                        LIST OF (Paragraph);
                                        LIST OF (Section);
                                        END;
                              END);
         END;
Paragraph = CASE OF
                    Simple-Paragraph = LIST OF (UNIT);
                    Displayed-Formula = Math-Formula;
                    END;
```

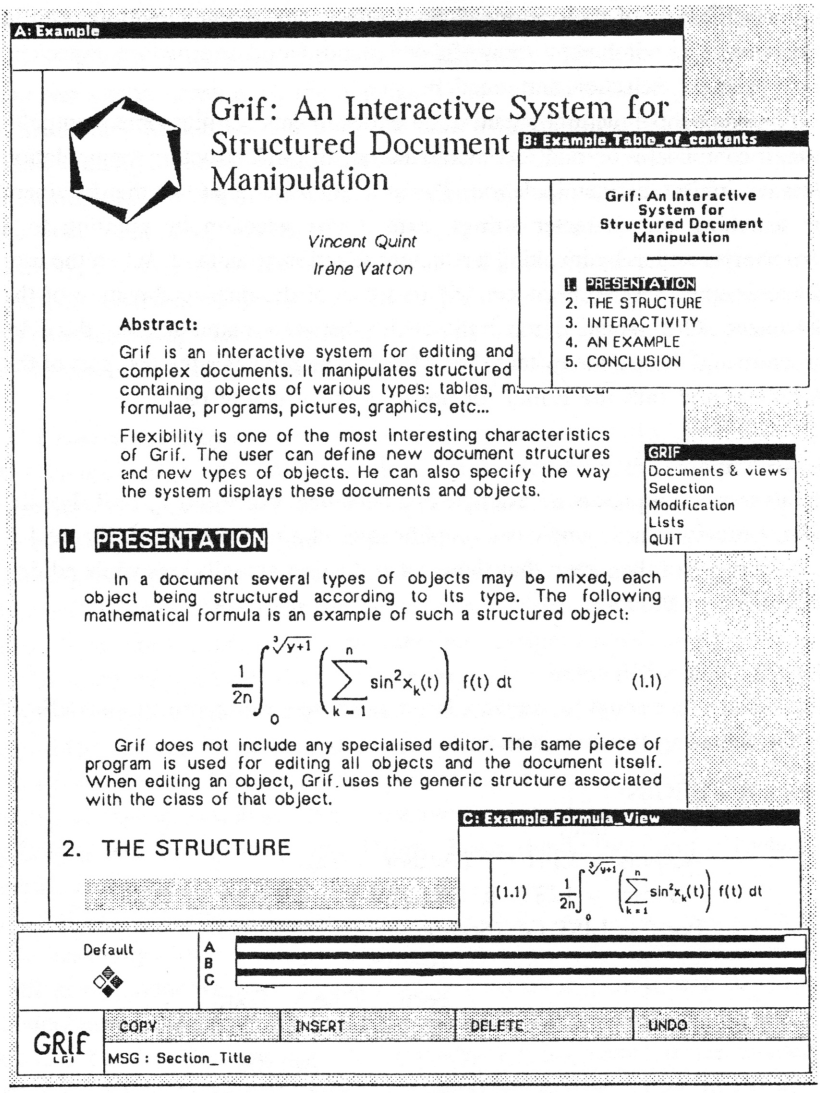
A: Example
Grif: An Interactive System for Structured Document Manipulation
Vincent Quint
Irène Vatton
Abstract:
Grif is an interactive system for editing and
complex documents. It manipulates structured
containing objects of various types: tables,
formulae, programs, pictures, graphics, etc...
Flexibility is one of the most interesting characteristics of Grif. The user can define new document structures and new types of objects. He can also specify the way the system displays these documents and objects.
1. PRESENTATION
In a document several types of objects may be mixed, each object being structured according to its type. The following mathematical formula is an example of such a structured object:
$$\frac{1}{2n}\int_0^{\sqrt[3]{y+1}}\left(\sum_{k=1}^{n}\sin^2 x_k(t)\right) f(t)\,dt \qquad (1.1)$$
Grif does not include any specialised editor. The same piece of program is used for editing all objects and the document itself. When editing an object, Grif uses the generic structure associated with the class of that object.
2. THE STRUCTURE
B: Example.Table_of_contents
Grif: An Interactive System for Structured Document Manipulation
1. PRESENTATION
2. THE STRUCTURE
3. INTERACTIVITY
4. AN EXAMPLE
5. CONCLUSION
GRIF
Documents & views
Selection
Modification
Lists
QUIT
C: Example.Formula_View
(1.1)
Default
A
B
C
GRIF
COPY
INSERT
DELETE
UNDO
MSG : Section_Title

Figure 1: A screen picture

At the first level, a Report is made up of a Title, Authors, an Abstract and a Body. The Title is just Text, i.e. a character string. The element Authors is a sequence of elements Author, each one being a character string. The Abstract is a sequence of elements Paragraph. The Body is a sequence of elements Section. A Section is made up of a Section-Title (a character string) and a Section-Body. The Section-Body contains a sequence of elements Paragraph followed by a sequence of lower level elements Section. At each level, a Section

has the same structure: a sequence of elements Paragraph followed by a sequence of lower level elements Section; the rule defining the structure of a Section being recursive.

A Paragraph may appear in the Abstract or in a Section. It is either a Displayed-Formula or a sequence of UNITs. A Displayed-Formula has the same structure as Math-Formula. As Math-Formula is not defined in this schema, it is assumed to be defined by another schema called Math-Formula, which defines the generic structure of mathemetical formulæ.

A UNIT is either a basic element or a structured element. Basic elements are Text (character string), Graphics (vector, rectangle, circle,...), or Pictures (bitmaps). So with this structure schema, it is possible to put a small picture within the text of a paragraph. As a UNIT may also be a structured element, it is also possible to put a formula within the text, or any other object of a class defined by another structure schema.

5.2. The Presentation

The presentation schema that has been used for editing this document contains presentation rules for each type of element defined in the structure schema. As it would be too long to show the program which defines these rules, we shall present here only some of them.

Three views are defined (figure 1), and all three have been created by the user. One view, called the *main view,* shows all elements of the document; it is always created by the system. Another view, named Table-of-contents, shows only the title of the report and the Section-Titles of the first level sections (displayed in window B). The last view, named Formula-View shows only Displayed-formulæ (displayed in window C). Different presentation rules have been specified for elements which are displayed in several views. For example, in Formula-View, Displayed-formulæ are written with smaller characters than in the main view and the formula number is not in the same place. The title of the report is centered in the Table-of-contents, and left aligned in the main view.

Several presentation elements are added by the presentation schema. These elements are related to elements of the structure and they are automatically created by the system when the element of the structure is created. As for any other element, the presentation schema defines the way to display them, but for these presentation elements it also defines their content which may be either constant or variable.

Constant elements may be any type of basic unit with a specified value. In this example three constant elements are used: the "Logo" (a Picture UNIT associated with the type Report) at the top left corner of the report, the word "Abstract:" (a Text UNIT associated with the type Abstract) at the top left

corner of the abstract, and the "Horizontal-Bar" (a Graphic UNIT associated with the type Title) centered under the title, but only in the view Table-of-contents.

Two types of variable presentation elements are used in this schema: section number and formula number. The section number is associated with the type Section-Title and its value is the rank of the section in the sequence of sections it belongs to. The formula number is associated with the type Displayed-Formula. Its value is the rank of the first level section in which the formula appears, followed by the rank of the displayed formula in the section.

Even empty elements are displayed in accordance with their presentation rules. In the example, one can see the first paragraph of section 2 as a grey rectangle having the width of any other paragraph.

Presentation rules may depend on the context. In our example, rules specify that the content of a Paragraph is split into lines, that these lines are justified, and that the width of the paragraph is the width of the enclosing element. Another rule specifies that the indentation of the first line of a Paragraph is inherited from the enclosing element. A rule for the type Abstract defines the value of the indentation as zero. Therefore the first line of paragraphs is not indented in the abstract. The type Section-Body has a presentation rule specifying a positive value for this indentation. Hence the difference between paragraphs in the abstract and in the sections.

5.3. Selection

Figure 1 shows the document when the user has selected the Section-Title of the first Section. When selecting, the type of the selected element is displayed at the bottom of the editor window (e.g. Section-Title at the bottom of the screen). In addition, the selected element is displayed in reverse video in all views in which it appears (e.g. in the main view and in the Table-of-contents).

When an element of the structure is selected, all presentation elements related to it are also selected. In figure 1, a section number has been automatically created by the presentation schema, therefore it is selected when the user clicks on the word "PRESENTATION".

It is possible to select any visible element using the mouse. So, if the user wants to modify the text of the conclusion, he may select this part by clicking on the word "CONCLUSION" in the Table-of-contents; the main view displays then the beginning of the conclusion.

5.4. Editing

When creating a report, Grif automatically creates the structure of the new document according to the generic structure. Thus it creates a Title, a first Author, an Abstract containing a Paragraph and a Body with only one Section

containing a Section-Title and a Section-Body (a first Paragraph and a first Section of level 2). All these elements are displayed as grey rectangles with the size and position specified in the presentation schema. For entering the text of an element, the user just has to click on its grey rectangle (its type then being displayed at the bottom of the editor window), and to type. While typing, the text is displayed according to the presentation schema: justified, or centered, and with the correct font.

In our example, when the first paragraph of section 1 is finished, the user presses a function key, 'Next', and a new Paragraph is created after the current one; this paragraph being automatically selected (the grey rectangle becomes dark). As the second paragraph will not be just text, he cannot type immediately. He clicks the command 'Insert' in the editor window and a menu is diplayed for choosing among the various possible contents of a paragraph (Simple-Paragraph or Displayed-Formula: this menu is built following the structure schema). He selects the entry 'Displayed-Formula' and an empty formula is created: a small grey rectangle centered in the window, with the number (1.1) on the right margin.

The structure schema of a formula specifies that a formula has the same structure as a mathematical expression which is a sequence of mathematical constructs. A construct is either a character string, a fraction, an integral, or a summation, etc... A fraction contains a numerator and a denominator, both being mathematical expressions. An integral contains an integration symbol (a single character) two limits (expressions) and the function to be integrated (an expression); the other constructs being defined in a similar way. This structure is very similar to that used by Edimath [Quint], therefore a formula is handled with Grif in the same way as in Edimath.

When the formula is created, a menu is displayed for selecting the type of its first construct. The user selects the entry 'Fraction'. The grey rectangle is then replaced by a small fraction bar with a grey rectangle above (the numerator) and another below (the denominator). The upper one is selected (in dark grey), the user types '1' and presses the 'End' function key which means that the numerator is finished (the 'Next' key would have created another construct within the numerator, after the character string '1'). The '1' replaces the grey numerator and the empty denominator is selected. The user types '2n' and presses 'End'. The fraction is now complete and another empty construct is created after the fraction. The 'construct' menu is displayed and the user selects 'Integral'. Four grey rectangles appear on the screen, for the integral symbol, the two limits and the function to be integrated. The user enters these elements in the same way as in the fraction. While entering the formula, the formula view is updated. The user may therefore act on either this view or the main view for editing the formula.

At any time, the user may suspend this sequential creation by clicking within any element. He can then edit this element and come back to the one he was previously creating.

6. Conclusion

Grif is implemented on a Perq running the PNX operating system, a version of UNIX. It uses all available features of the PNX window manager and its graphics primitives.

Grif is a part of a larger project, called Tigre, which aims at developing a complete environment for documents. Another important part of Tigre is the generalised data base management system [Lopez] which uses the same notion of structured documents [Bogo]. So, documents produced with Grif can be stored in the data base and their structure may be used when they are processed by the data base system.

Further developments of Grif are intended, specifically an adaptation of the Mediator for simple terminals. This will allow us to use the full flexibility of the architecture.

References

[1] André, J., Grundt, Y., Quint, V. (1985). Towards an interactive math mode in TEX. In *TEX for Scientific Documentation*, ed. D. Lucarella, pp. 79-92, Addison-Wesley.

[2] Bogo, G., Richy, H., Vatton, I. (1983). Un modèle de représentation de documents généralisés. In *Actes des journées sur la manipulation de documents*, ed. J. André, pp. 221-235, INRIA, Rennes.

[3] Chamberlin, D.D., King, J.C., Slutz, D.R., Todd, S.J.P., Wade, B.W. (1981). JANUS: An Interactive System for Document Composition. In *ACM SIGPLAN Notices*, **16**, 6.

[4] Donzeau-Gouge, V., Kahn, G., Lang, B., Mélèse, B., Morcos, E. (1983). Outline of a tool for document manipulation. In *IFIP 83*, pp. 615-620, Paris, R.E.A. Mason ed., Elsevier Science Publishers B.V.

[5] Estublier, J., Krakowiak, S., Mossière, J., Rouzaud, Y. (1983). Design Principles of the Adèle Programming Environment. In *International Computing Symposium on Application Systems Development*, ACM.

[6] Furuta, R., Scofield, J., Shaw, A. (1982). Document Formatting Systems: Survey, Concepts, and Issues. *Computing Surveys*, **14**, 3, pp. 417-472.

[7] Gosling, J. (1984). An Editor Based User Interface Toolkit. In *Protext I*, ed. J.J.H. Miller, Dublin, Boole Press.

[8] Gutknecht, J., and Winiger, W. (1984). Andra: The Document Preparation System of the Personal Workstation Lilith. *Software-Practice and Experience*, **14**, pp. 73-100.

[9] Harslem, E., Nelson, L.E. (1982). A Retrospective on the Development of Star. In *Proceedings of the 6th International Conference on Software Engineering*, pp. 377-383, Tokyo.

[10] Hibbard, P. (1983). *User Manual for Mint - The Spice Document Preparation System*. Spice Document S153, Carnegie-Mellon University.

[11] Joloboff, V. (1983). An Interactive Graphics Editor for Document Preparation. In *Proceedings of 1983 ACM Conference on Personal and Small Computers*, pp. 45-53, San Diego.

[12] Joloboff, V., Quint, V. (1984). Two-dimensional editing, In *Proceedings of the IEEE First International Conference on Office Automation*, pp. 41-49, New Orleans.

[13] Kasik, David J. (1982) A User Interface Management System, In *Siggraph'82 Conference Proceedings*, ed. R. Daniel Bergeron, pp. 99-106, *Computer Graphics*, **16**, 3.

[14] Kernighan, B.W. (1984). The Unix Documentation Preparation Tools - A Retrospective. In *Protext I*, ed. J.J.H. Miller, Dublin, Boole Press.

[15] Knuth, D.E. (1984). *The TEXbook*, Addison-Wesley, Reading, Mass.

[16] Lopez, M., Palazzo Oliveira, J., Velez, F. (1983). The TIGRE Data Model, *Rapport de Recherche TIGRE* n.2, IMAG, Grenoble.

[17] Medina-Mora, R. (1982). *Syntax directed editing: toward integrated programming environments*, CMU-CS-82-113, Carnegie-Mellon University.

[18] Meyrowitz, N., van Dam, A. (1982). Interactive Editing Systems, *Computing Surveys*, **14**, 3, pp. 321-415.

[19] Quint, V. (1983). An interactive system for mathematical text processing, *Technology and Science of Informatics*, **2**, 3, pp. 169-179.

[20] Reid, B.K. (1980). A high-level approach to computer document production. In *Proceedings of the 7th Annual ACM Symposium on Principles of Programming Languages*, ACM SIGPLAN-SIGACT.

[21] Reiss, S.P. (1984). Graphical Program Development with PECAN Program Development Systems. In *ACM Sigplan Notices*, **19**, 5.

[22] Teitelbaum, T., Reps, T. (1981). The Cornell program synthesizer: a syntax directed programming environment. *Communications of the ACM*, **24**, 9, pp. 563-573.

[23] Wong, Peter C. S., Reid, Eric R. (1982). Flair - User Interface Dialog Design Tool, In *Siggraph'82 Conference Proceedings*, ed. R. Daniel Bergeron, *Computer Graphics*, **16**, 3, pp. 87-98.